Matrix Multiplication

On CCR Cluster

CSE 633 Parallel Algorithms

Spring 2014

Praveen Kumar Bellam

Outline

- Sequential Algorithm
- Parallel Algorithm
- Parallel Implementation Using MPI
- Parallel Implementation Using Open MP
- Results

Input: Two square matrices A, B *of size* $n \times n$
Output: The matrix product $C_{n \times n} = A \times B$

Matrix Multiplication (A, B)

```
for i = 1 to n do
   for j = 1 to n do
    c_ij = 0
    for k = 1 to n do
      c_ij = c_ij + a_ik b_kj
    end for
  end for
end for
```

$\Theta(n^3)$

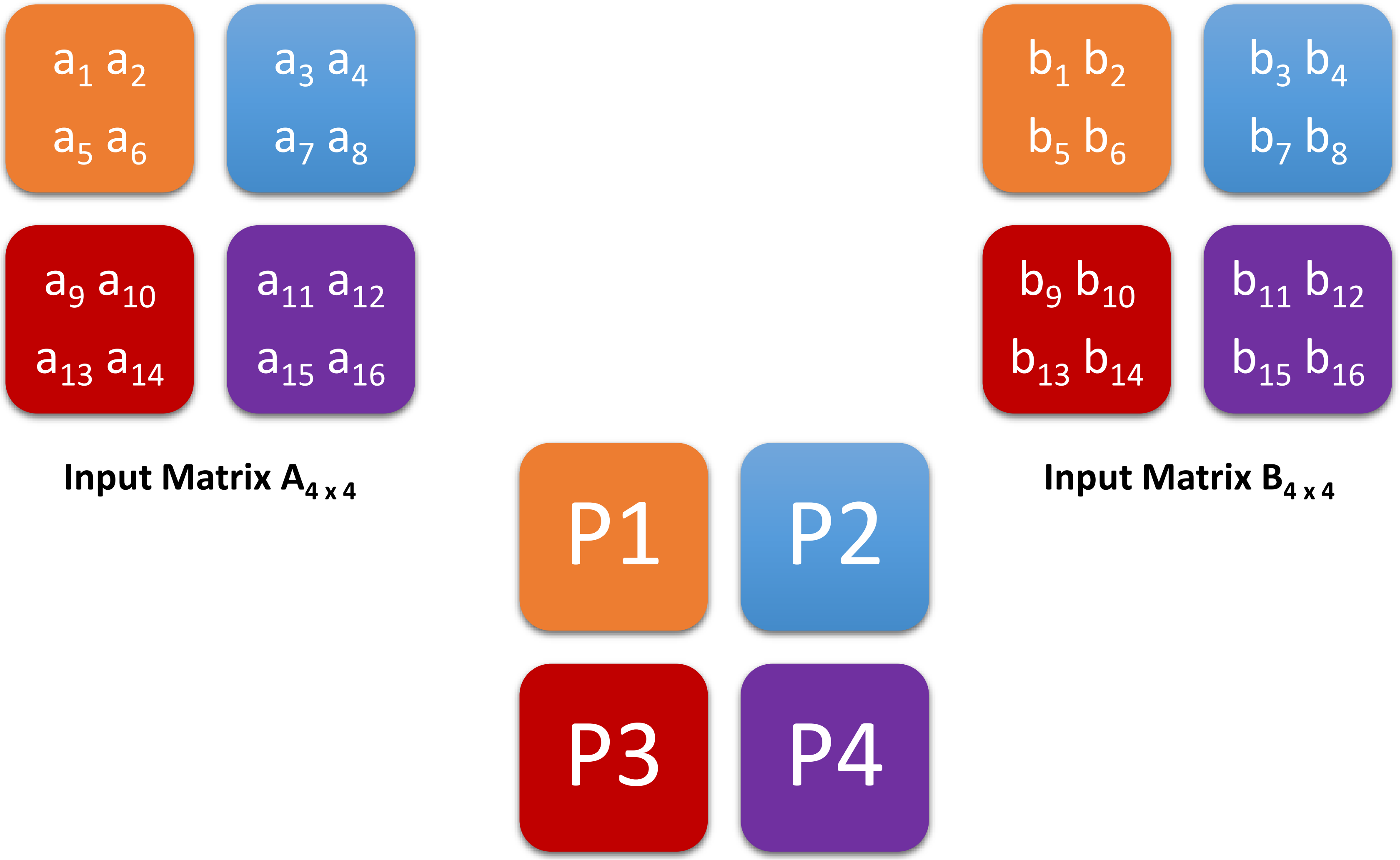
Parallel Algorithm – Design Considerations

- Distributing the data
- Local computation
- Communicating the data (Send/Receiving)
- Gathering the results

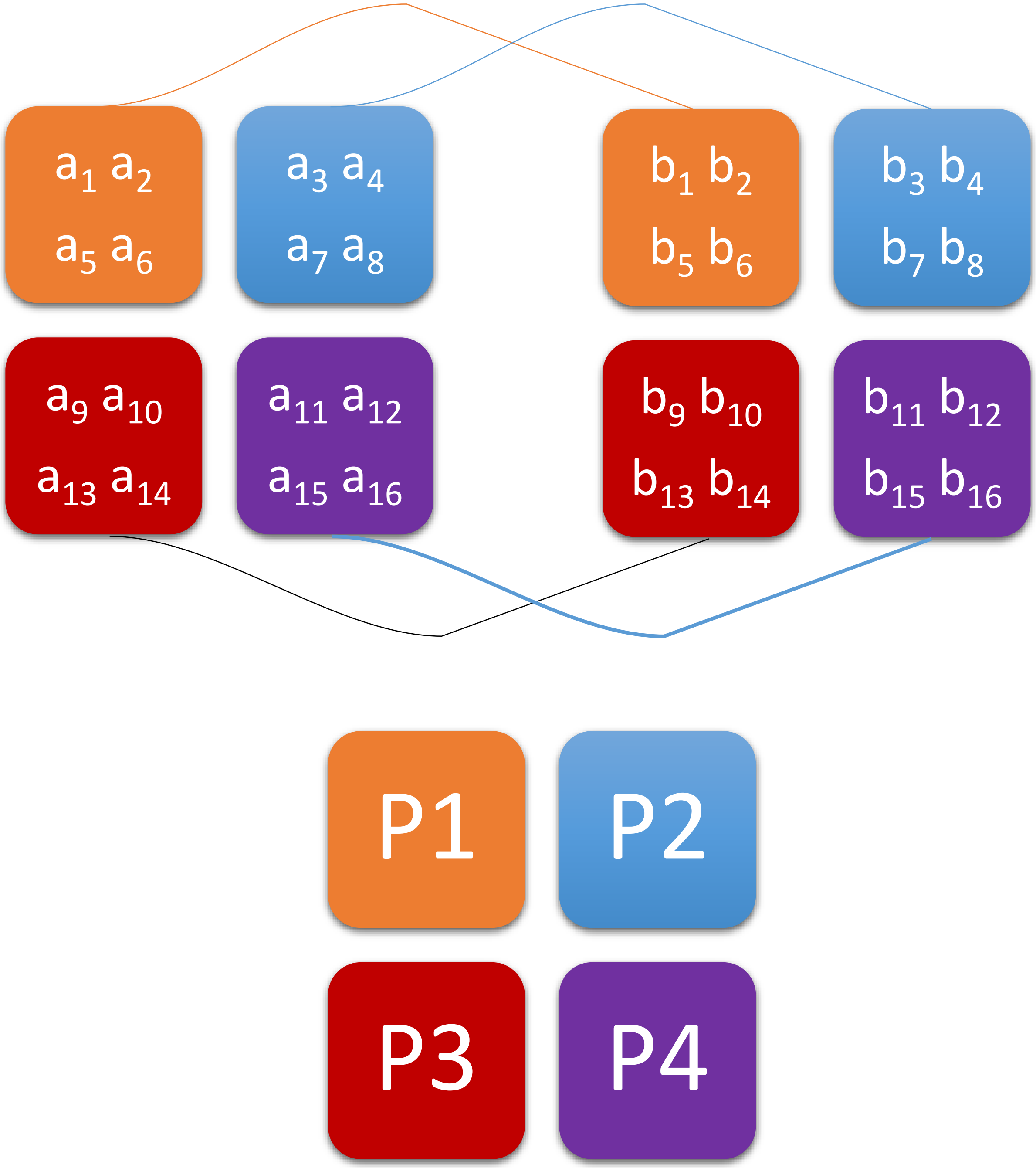
Parallel Algorithm

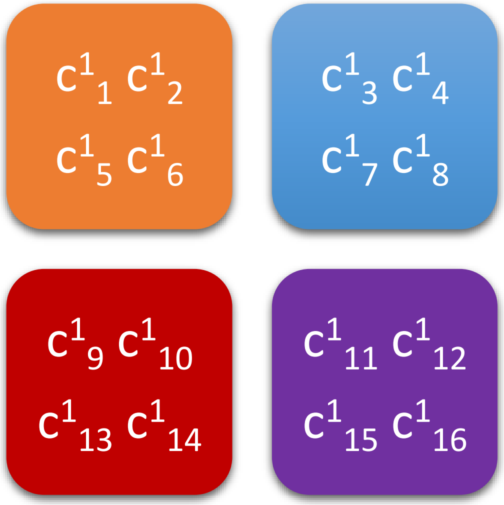
1. Divide the input matrices into P blocks. P is the number of processors available for computation.
2. Create a matrix of processes of size $P^{1/2}$ x $P^{1/2}$ so that each process can maintain a block of A matrix and a block B matrix.
3. Each block is sent to each process by determining the owner, and the copied sub blocks are multiplied together and the results added to the partial results in the C sub-blocks.
4. The A sub-blocks are rolled one step to the left and the B sub-blocks are rolled one step upwards.
5. Repeat the process, $P^{1/2}$ times.

Divide input matrices into P sub blocks, and distribute the data

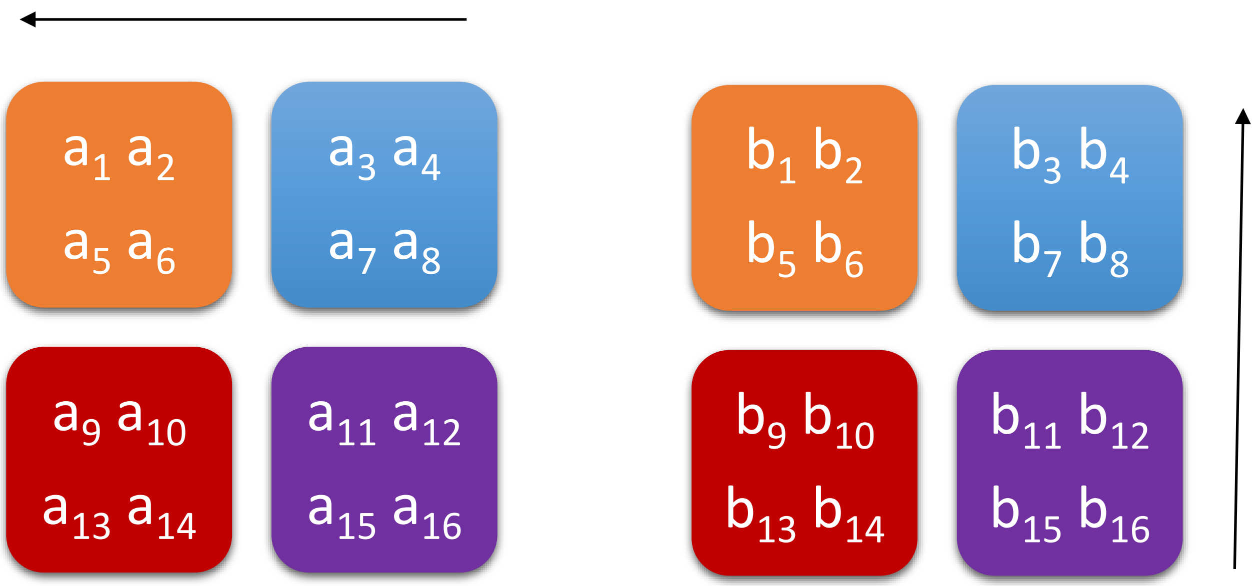


Each processor, performs local matrix multiplication of blocks.



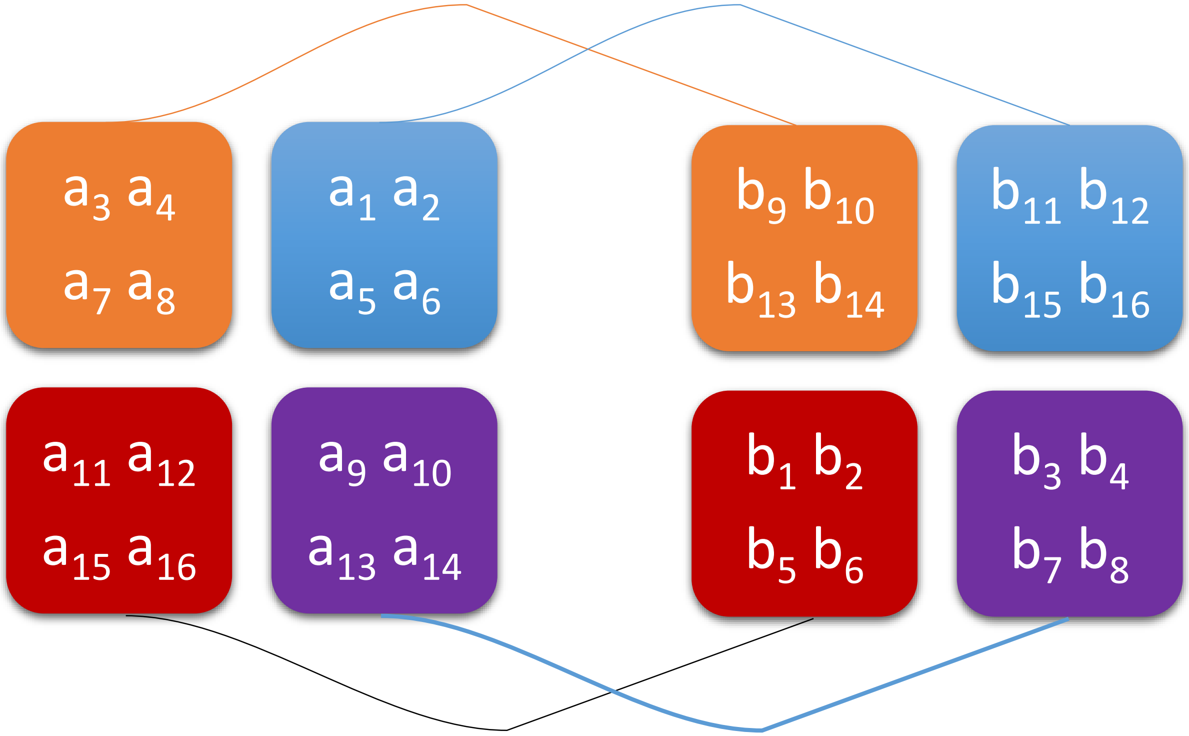


Output Matrix $C_{4 \times 4}$

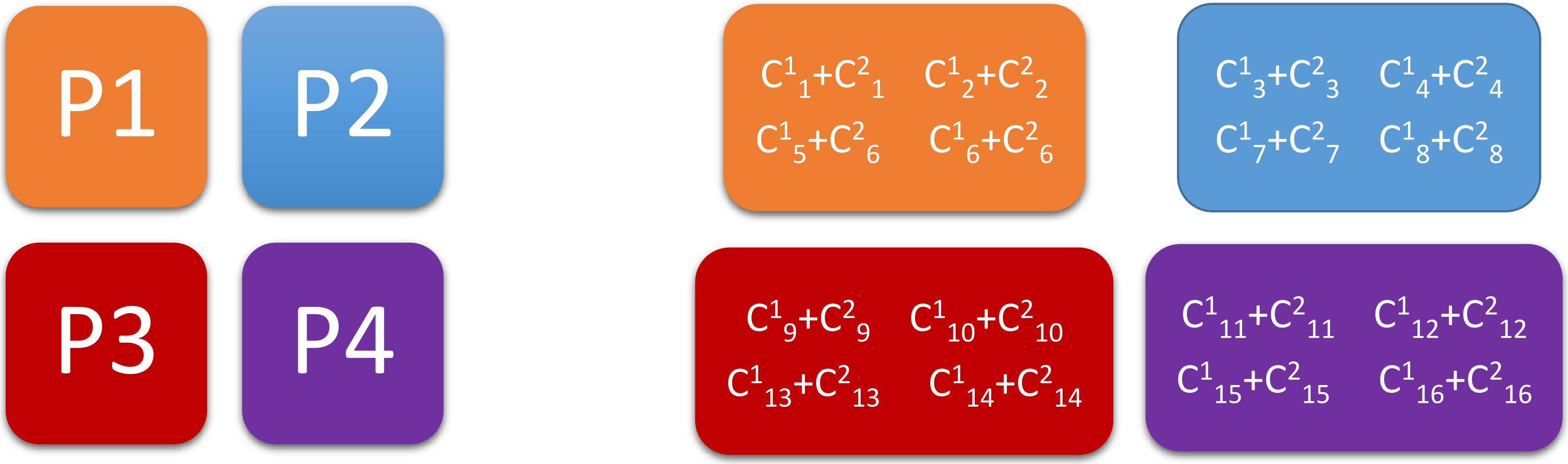


Each processor, sends A's sub block to the processor on the left, B's sub block to the processor above.

Output Matrix $C_{4 \times 4}$



Again, perform local matrix multiplication and add it to the result set.

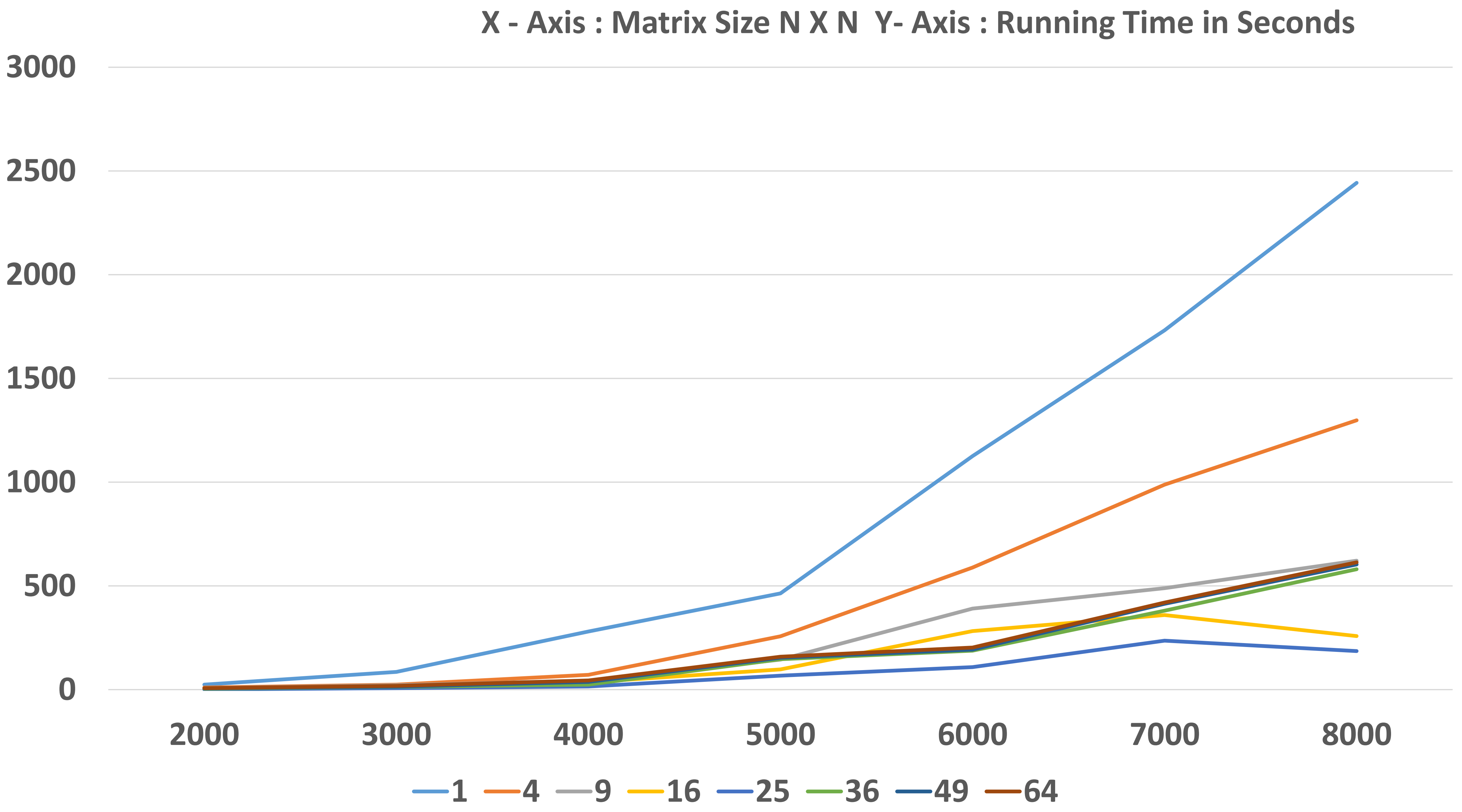


Matrix Size →

MPI Results

NUMBER OF PROCESSORS ↓

	2000	3000	4000	5000	6000	7000	8000
1	24	85	280	464	1125	1732	2443
4	9.43	23.70	70.90	256.62	588.38	988.20	1297.69
9	6.85	20.50	42.56	144.59	390.40	488.72	620.96
16	5.20	15.10	35.16	97.14	281.76	359.46	257.72
25	1.33	6.80	14.79	67.08	107.91	235.77	186.07
36	4.20	12.70	24.01	146.51	187.40	380.78	580.31
49	5.72	13.78	36.78	153.08	192.66	413.12	603.64
64	6.80	18.30	44.43	158.65	202.74	419.35	613.37


X - Axis : Matrix Size N X N Y- Axis : Running Time in Seconds
3000
2500
2000
1500
1000
500
0
2000
3000
4000
5000
6000
7000
8000
1
4
9
16
25
36
49
64

OpenMP Results

Matrix Size →

NUMBER OF PROCESSORS ↓

	2000	3000	4000	5000	6000
1	24	85	280	464	1125
2	11.36	44.35	132.12	267.05	403.06
4	8.23	31.66	99.08	206.26	362.38
6	6.57	23.43	71.32	139.01	253.85
8	4.44	15.66	57.31	115.05	204.02
10	3.26	9.30	45.54	88.13	162.54
12	2.44	7.30	37.75	86.69	127.68
14	2.37	6.02	25.47	52.52	109.34

Observations & Learnings

- Increasing the processors doesn't always reduce the running time.
- Need to experimentally identify the point where communication costs are taking over the local computation.
- Running times depend on how the nodes got allocated on CCR cluster.
- Need to specify in the SLURM Script about the node details.
- Got good understanding about parallelization.
- Next step is to analyze and understand the semantics of parallel architectures practically by simulating the algorithms.

References

- Gupta, Anshul; Kumar, Vipin; , "Scalability of Parallel Algorithms for Matrix Multiplication," *Parallel Processing, 1993. ICPP 1993. International Conference on* , vol.3, no., pp.115-123, 16-20 Aug. 1993 doi: 10.1109/ICPP.1993.160 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4134256&isnumber=4134231